Bilgisayar Mühendisliğine Giriş – BLM 101

Hafta 11: FB-CPU

Fenerbahçe Üniversitesi

10. Hafta İçeriği

- FB-CPU Nedir?
- Kullanılacak Araçlar
- Saklayıcılar
- Bellek
- İşlem Ünitesi
- Kontrol Ünitesi
- FB-CPU Makine Dili
- Proje Teslimi

FB-CPU

- FB-CPU işlemcilerin temel çalışma prensiplerini anlatmak için, eğitim amaçlı bir işlemcidir.

- Von Neumann mimarisi ile tasarlanmıştır.

FB-CPU

İşlemci;

- Bellek (RAM)
- Saklayıcılar
- Kontrol Ünitesi
- Aritmetik İşlem Ünitesi

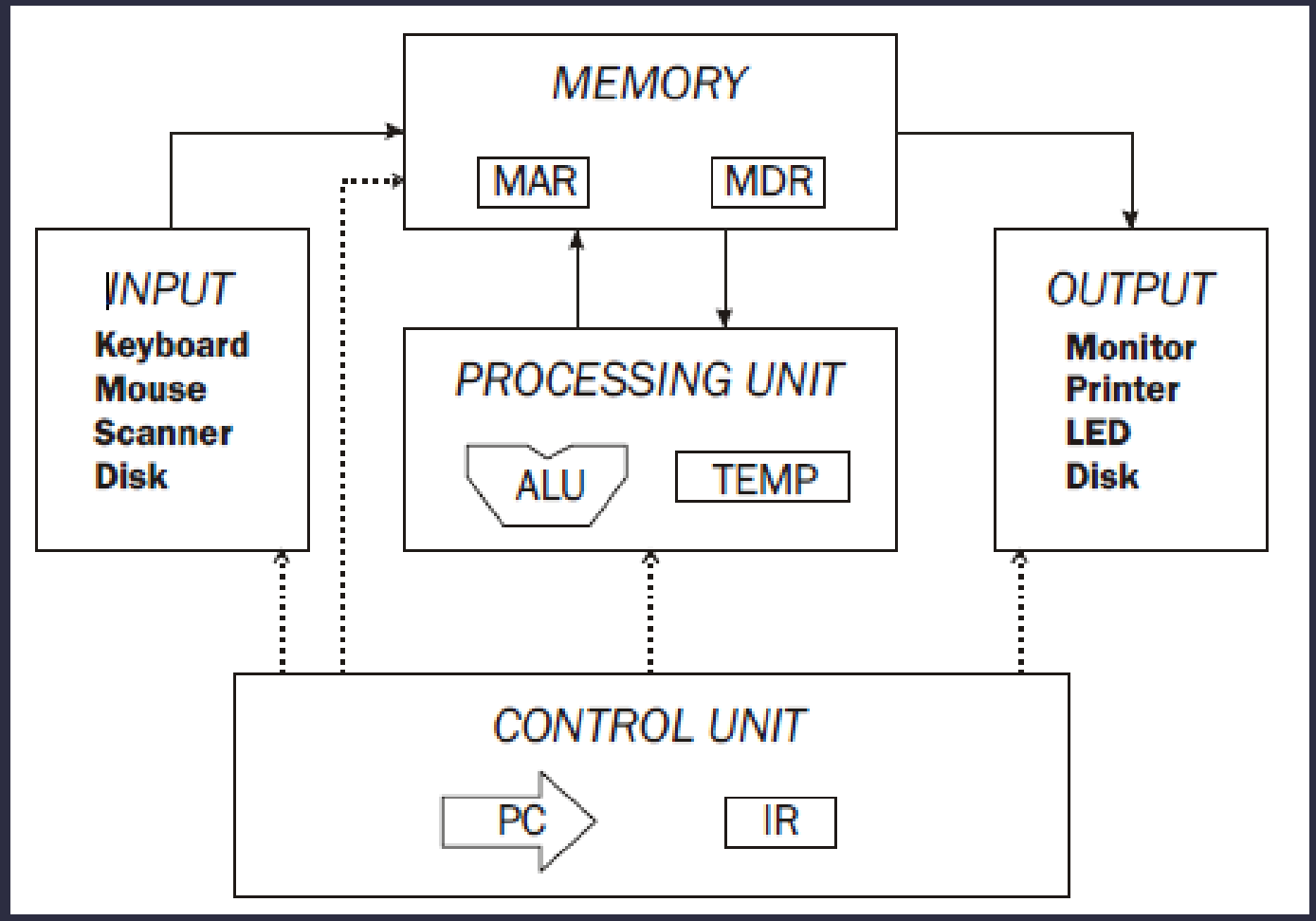
yapılarını içermektedir.

FB-CPU

İşlemci tasarlanırken kullanılacak araçlar;

- Von Neumann Simulatorü
  - İşlemci için algoritma geliştirmede faydalıdır
- Logisim-Evolution
  - İşlemcinin devresinin tasarlanmasında kullanılacaktır

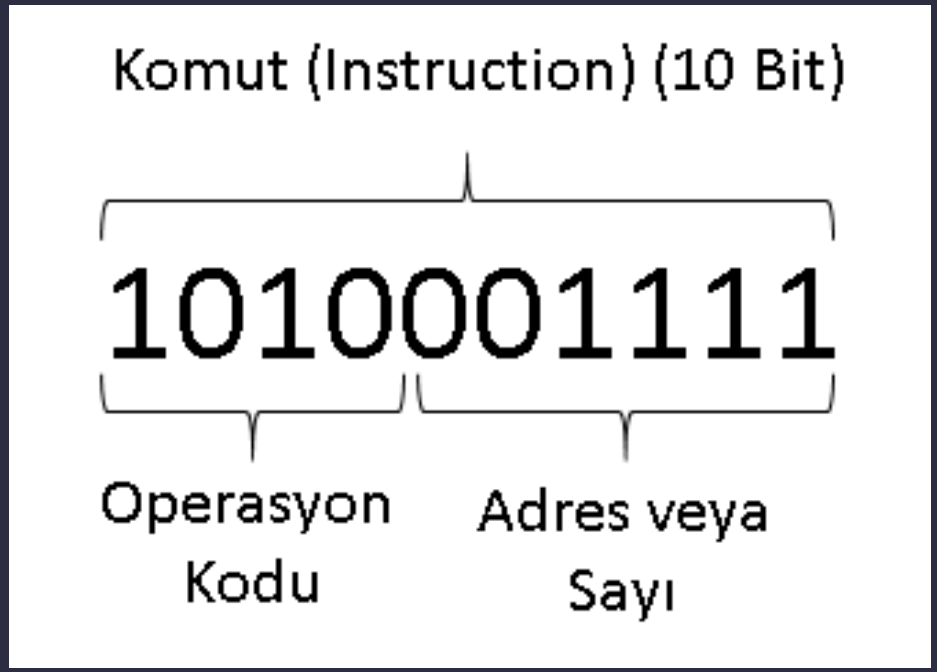
FB-CPU


MEMORY
MAR
MDR
INPUT
Keyboard
Mouse
Scanner
Disk
PROCESSING UNIT
ALU
TEMP
OUTPUT
Monitor
Printer
LED
Disk
CONTROL UNIT
PC
IR

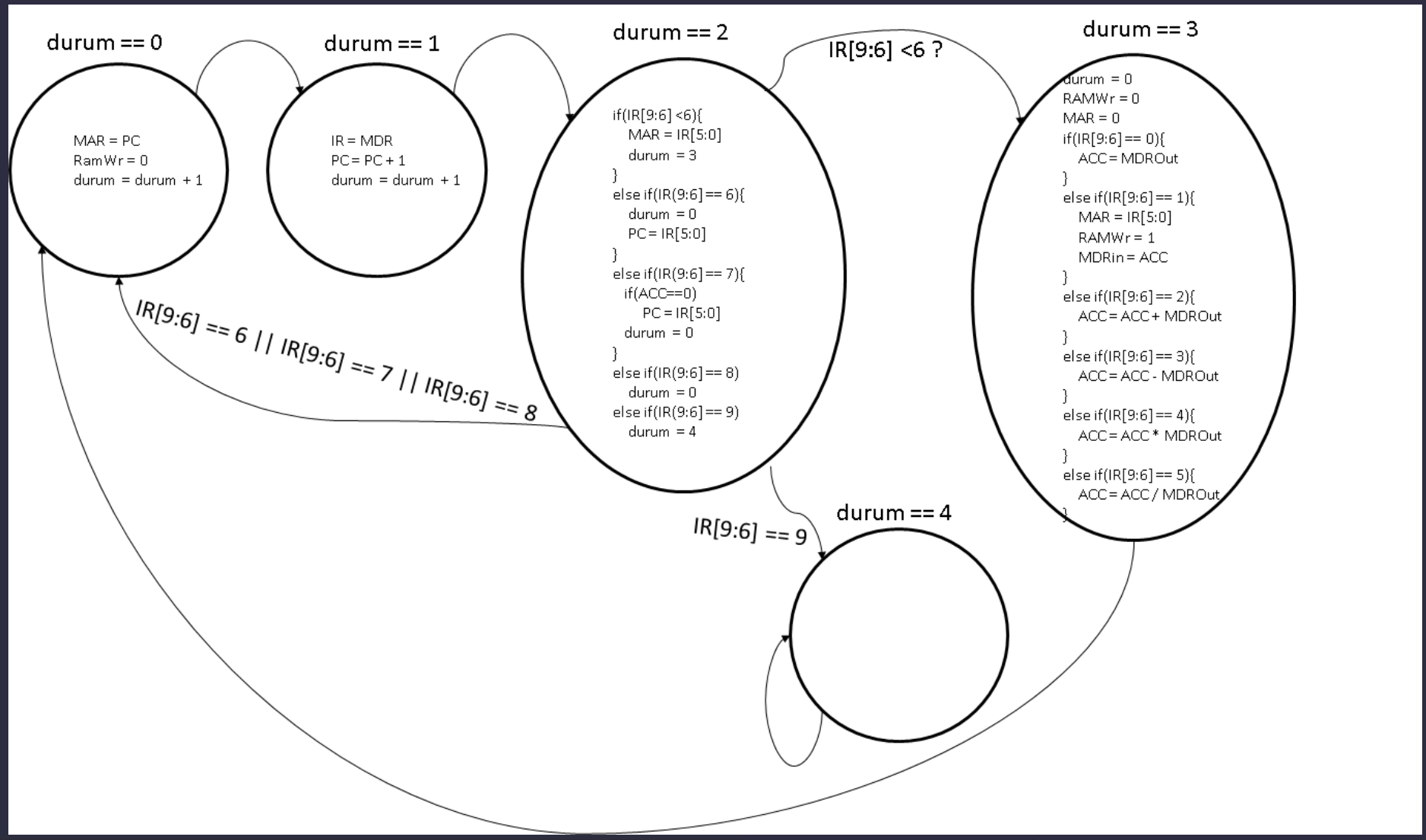
FB-CPU

Tablo 1. FB-CPU ISA (Instruction Set Architecture)

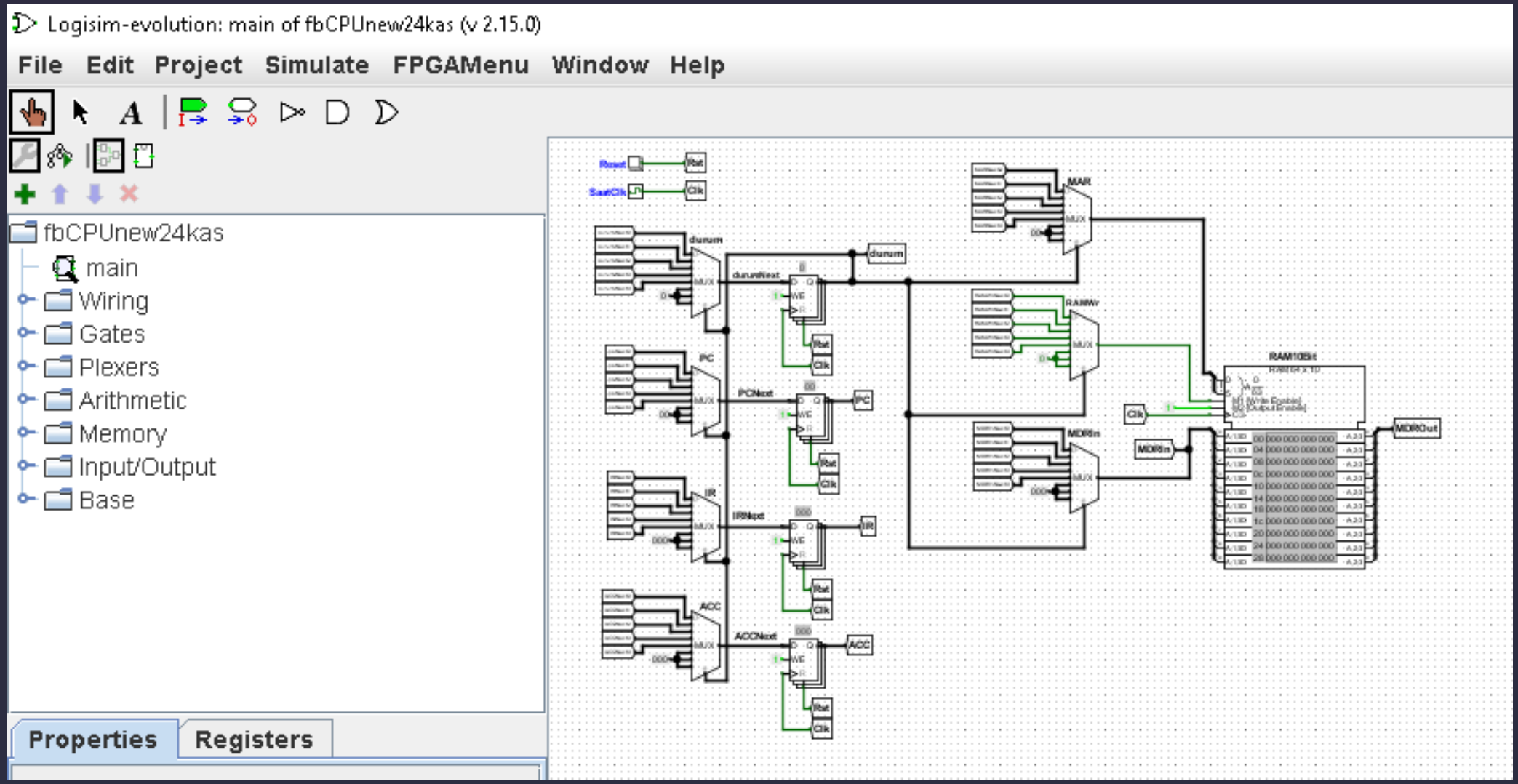
Komut Adı	Görevi	Operasyon Kodu
LOD ADDR	Yükleme (Load), Bellekteki verilen adresin içerisinden değeri alıp, ACC saklayıcısına yerleştirir. ACC = *(ADDR)	0000
STO ADDR	Kaydetme (Store), ACC'nin içerisindeki değeri alıp, bellekte verilen adrese yazar. *(ADDR) = ACC	0001
ADD ADDR	Bellekteki verilen adresteki değeri alır, ACC ile toplayıp, ACC'nin üzerine yazar. ACC = ACC +*(ADDR)	0010
SUB ADDR	Bellekteki verilen adresteki değeri alır, ACC ile çıkartıp, ACC'nin üzerine yazar. ACC = ACC - *(ADDR)	0011
MUL ADDR	Bellekteki verilen adresteki değeri alır, ACC ile çarpıp, ACC'nin üzerine yazar. ACC = ACC * (*(ADDR))	0100
DIV ADDR	Bellekteki verilen adresteki değeri alır, ACC ile bölüp, ACC'nin üzerine yazar. ACC = ACC / (*(ADDR))	0101
JMP SAYI	PC = Sayı olur.	0110
JMZ SAYI	ACC'ın değeri 0 ise, verilen sayı değerini PC'e atar, değilse işlem yapmaz.	0111
NOP	No Operation, hiçbir işlem yapılmaz.	1000
HLT	Uygulama durur	1001



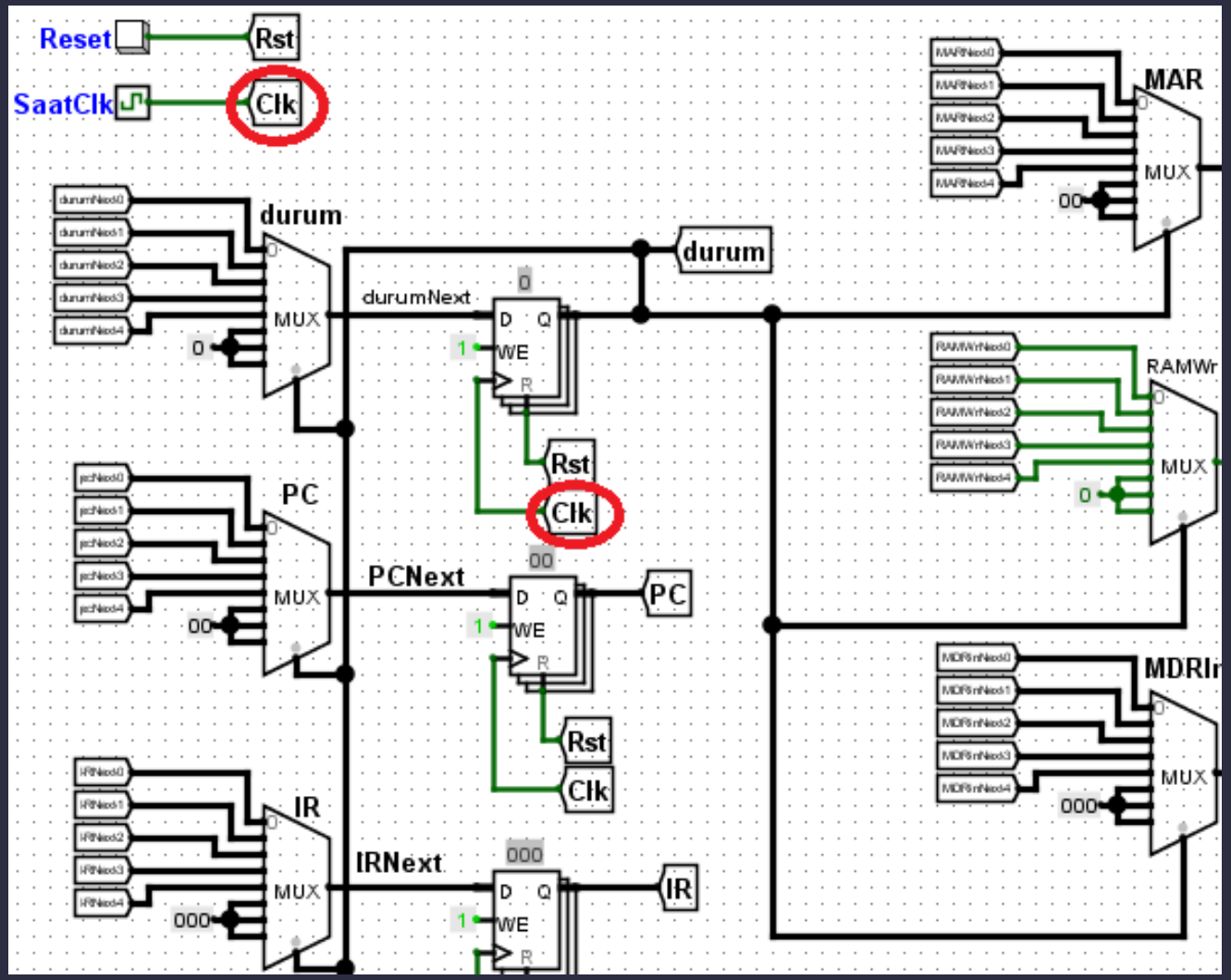
FB-CPU


durum == 0
MAR = PC
RamWr = 0
durum = durum + 1
durum == 1
IR = MDR
PC = PC + 1
durum = durum + 1
durum == 2
if(IR[9:6] <6){
MAR = IR[5:0]
durum = 3
}
else if(IR[9:6] == 6){
durum = 0
PC = IR[5:0]
}
else if(IR[9:6] == 7){
if(ACC==0)
PC = IR[5:0]
durum = 0
}
else if(IR[9:6] == 8)
durum = 0
else if(IR[9:6] == 9)
durum = 4
IR[9:6] <6 ?
durum == 3
durum = 0
RAMWr = 0
MAR = 0
if(IR[9:6] == 0){
ACC = MDROut
}
else if(IR[9:6] == 1){
MAR = IR[5:0]
RAMWr = 1
MDRin = ACC
}
else if(IR[9:6] == 2){
ACC = ACC + MDROut
}
else if(IR[9:6] == 3){
ACC = ACC - MDROut
}
else if(IR[9:6] == 4){
ACC = ACC * MDROut
}
else if(IR[9:6] == 5){
ACC = ACC / MDROut
}
IR[9:6] == 6 || IR[9:6] == 7 || IR[9:6] == 8
IR[9:6] == 9
durum == 4

FB-CPU


Logisim-evolution: main of fbCPUnew24kas (v 2.15.0)
File Edit Project Simulate FPGAMenu Window Help
fbCPUnew24kas
main
Wiring
Gates
Plexers
Arithmetic
Memory
Input/Output
Base
Properties
Registers
Reset
SaatClk
durum
durumNext
PC
PCNext
IR
IRNext
ACC
ACCNext
MAR
RAMWr
MDRIn
MDROut
RAM1Bit

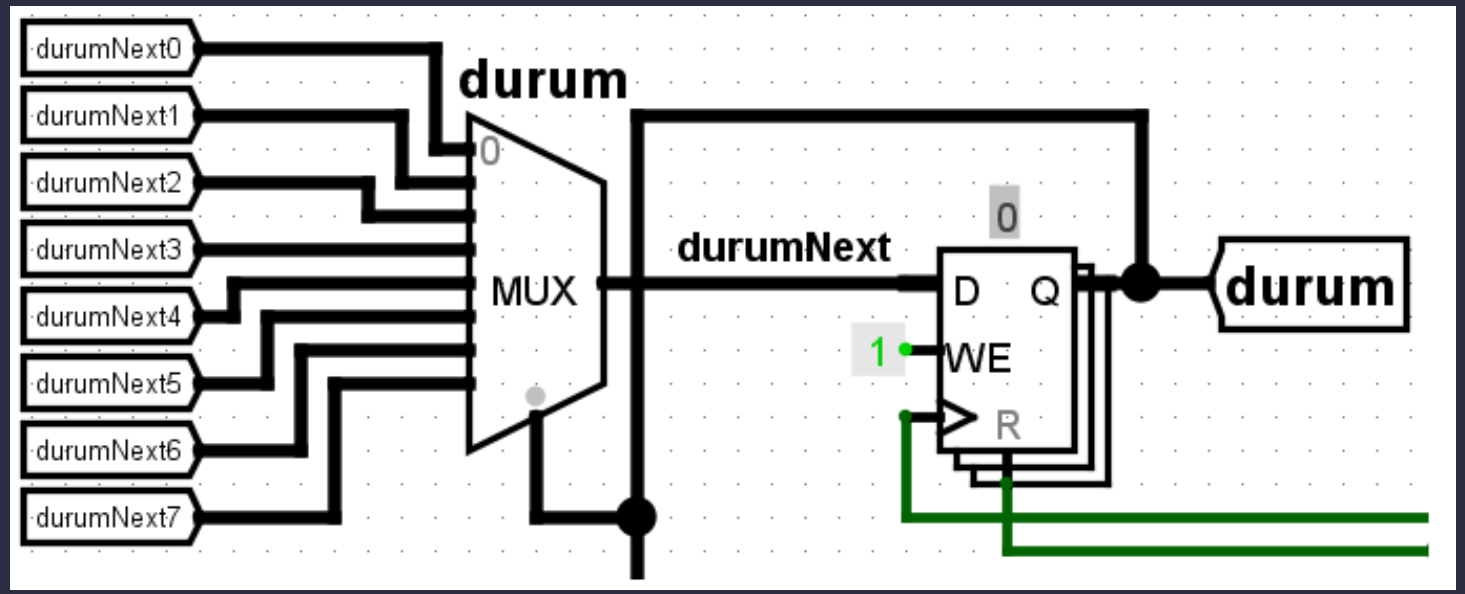
FB-CPU


Reset
Rst
SaatClk
Clk
durum
MUX
durumNext
D Q
WE
R
PC
PCNext
IR
IRNext
MAR
RAMWr
MDRIr
000
00
0
1

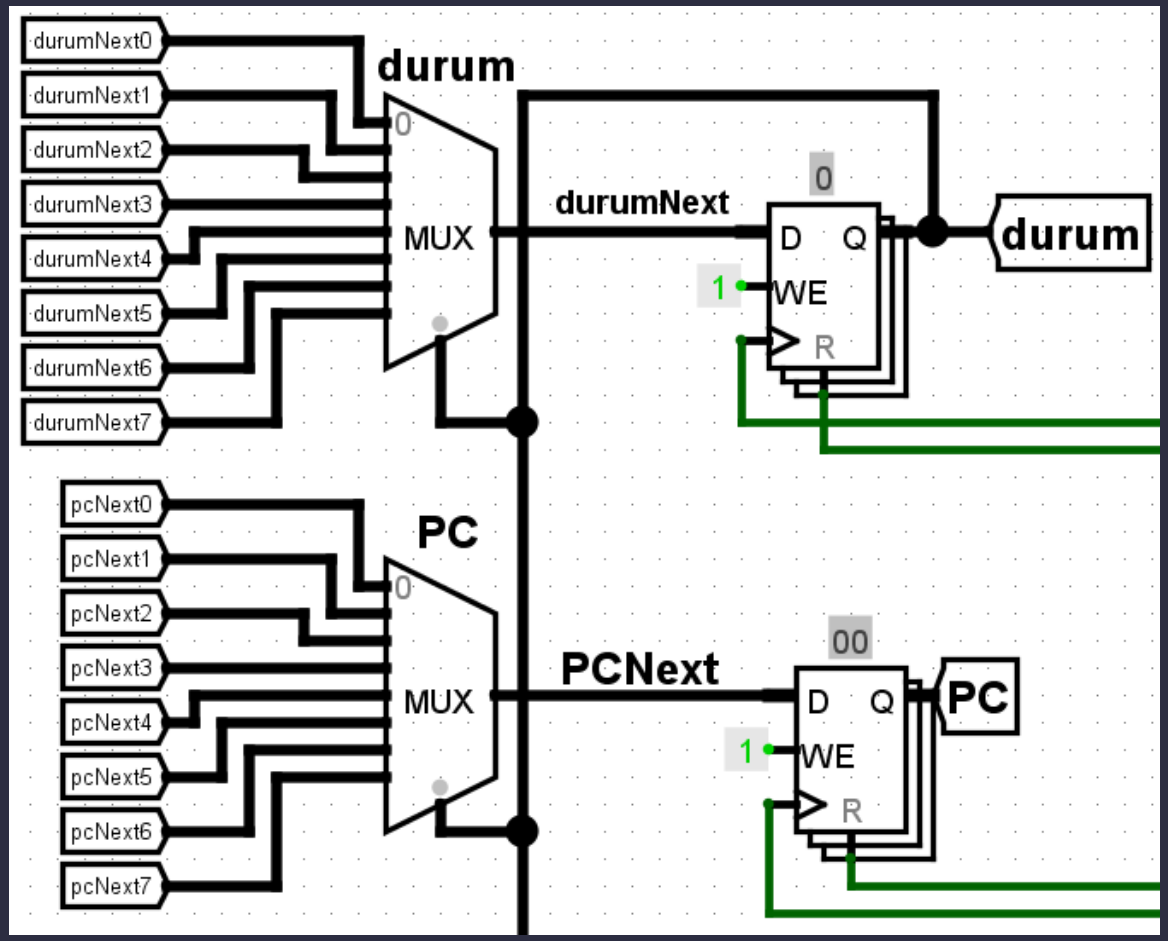
FB-CPU

Saklayıcılar

- **Durum (3 Bit)**: FB-CPU durum makinaları yöntemi ile gerçeklenecektir. Yani bu işlemci durum ismindeki saklayıcının değerine göre 2^3 = 8 farklı durumda çalışan bir tasarımı olacaktır (İşlemcinin desteklemesi istenen işlemlerin tamamı 8 farklı durumda yapılabilmektedir). Aşağıdaki şekile bakıldığında durum saklayıcısını ve kendisine bağlı olan MUX yapısı görülmektedir.



FB-CPU


durumNext0
durumNext1
durumNext2
durumNext3
durumNext4
durumNext5
durumNext6
durumNext7
durum
0
MUX
durumNext
0
D
Q
1
WE
R
durum
pcNext0
pcNext1
pcNext2
pcNext3
pcNext4
pcNext5
pcNext6
pcNext7
PC
0
MUX
PCNext
00
D
Q
1
WE
R
PC

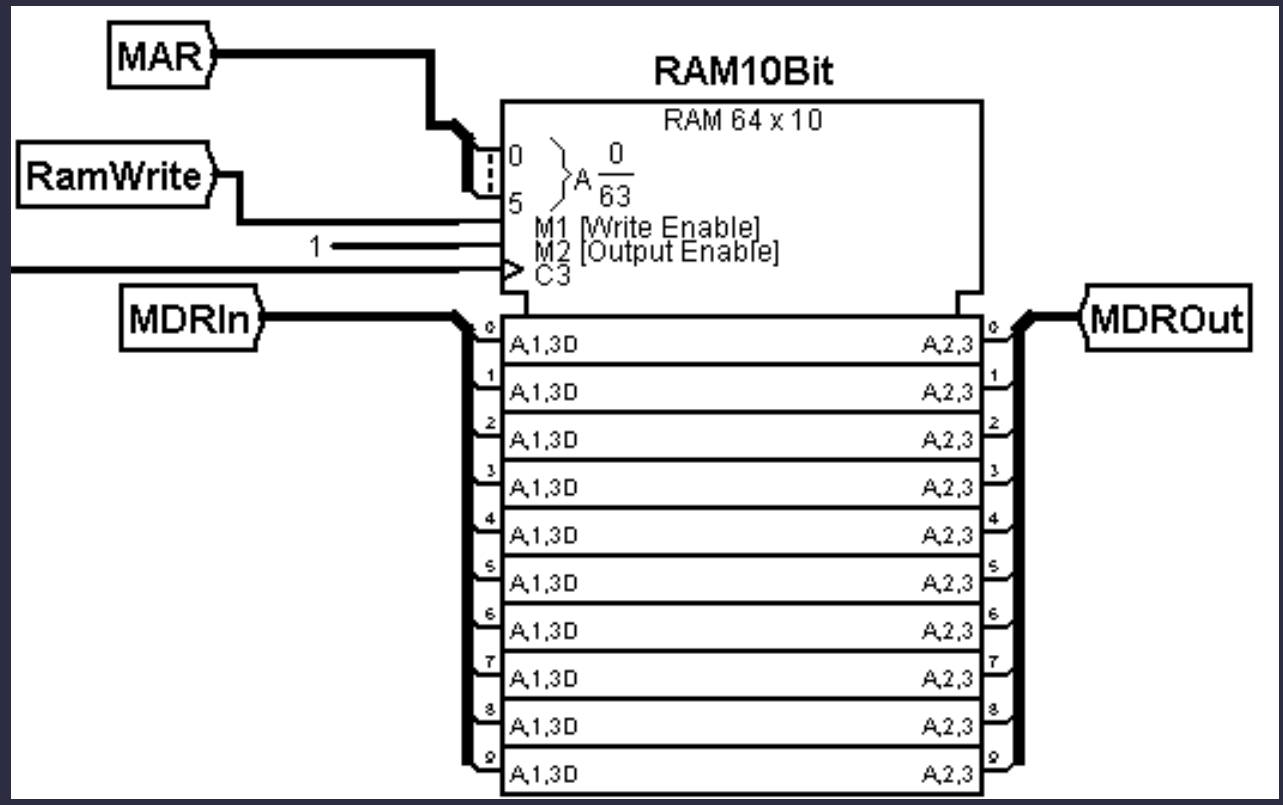
FB-CPU

Saklayıcılar

- **PC (6 Bit):** RAM üzerinde hangi satırdaki komutun alınacağını belirler. 6 bit olmasının nedeni RAM'in 2^6 lokasyonu olmasındandır. Dolayısıyla PC değeri RAM'deki her yeri gösterebilmektedir.
- **MAR (6 Bit):** Memory Address Register isminde bir saklayıcıdır. Bu saklayıcı RAM'in adres girişine bağlanmıştır. RAM'in 2^6 lokasyonu olduğu için MAR 6 bitliktir. Saklayıcı RAM'in içerisindedir.
- **MDRIn (10 Bit):** Memory Data Register In, RAM'e bir veri yazılacağı zaman kullanılan saklayıcıdır. RAM'in bir lokasyonu 10 bitlik olmasından ötürü, saklayıcı 10 bittir. Saklayıcı RAM'in içerisindedir.
- **RAMWr (1 Bit):** RAM'e veri yazılacağı durumlarda aktif edilmektedir. 1 olmadığı durumlarda RAM'e veri yazılmaz. Saklayıcı RAM'in içerisindedir.
- **MDROut (10 Bit):** Memory Data Register, RAM'den veri okunacağı zaman kullanılan saklayıcıdır. RAM'in bir lokasyonu 10 bit olmasından dolayı, saklayıcı 10 bittir. Saklayıcı RAM'in içerisindedir.
- **IR (10 Bit):** Instruction Register, RAM'den okunan kodun (instruction) saklandığı saklayıcıdır.
- **ACC (10 Bit):** Accumulator, aritmetik işlem sonuçlarının tutulduğu saklayıcıdır.

Bellek

FB-CPU'nun komutları okuyup, hesaplanan değerleri geri yazacağı bellek yandaki şekilde verilmektedir. RAM'e bağlı 4 saklayıcı ve bir clock sinyali bulunmaktadır. RAM'e bağlı saklayıcıların görevleri saklayıcılar bölümünde açıklanmıştır.



FB-CPU

- **İşlem Ünitesi (ALU, Arithmetic Logic Unit):**

Aritmetik işlemlerin gerçekleştirildiği bölümdür. FB-CPU'da 4 adet aritmetik işlem vardır. Bunlar toplama, çıkartma, çarpma ve bölmedir, gelen operasyon koduna göre işlemleri gerçekleştirip ACC saklayıcısına yazmaktadır.

- **Kontrol Ünitesi:**

Saklayıcılar, Aritmetik İşlem Ünitesi ve RAM'e verilerin birbirleri arasında transferinden sorumludurlar. İşlemci içi veri akışını yönetir.

Eksik Ünitelerin Tasarımı

- Verilen başlangıç tasarımında durum 0 ve durum 1 için tasarım verilmiştir.
- LAB'da durum 2'nin tasarımı yapılacaktır.
- İşlemcinin çalışır olması için durum 3'ün tamamlanması gerekmektedir.

FB-CPU

Örnek Yazılım 1

FB-CPU için bellekte 50 ve 51 adresteki iki sayının toplamını 52 no'lu adrese kaydeden uygulamayı geliştiriniz.

- 0: 0000_110010 // LOD 50, (ACC = *50), Hex = 32
- 1: 0010_110011 // ADD 51, ACC = ACC + (*51), Hex = B3
- 2: 0001_110100 // STO 52, (*52) = ACC, Hex = 74
- 3: 1001_000000 // Halt, Hex = 240
- 50: 0000000101 // Hex = 5
- 51: 0000001010 // Hex = A

FB-CPU

Örnek Yazılım 2

FB-CPU için bellekte 50 ve 51 adresteki iki sayının çarpımını 52 no'lu adrese kaydeden uygulamayı geliştiriniz.

- 0: 0000_110010 // LOD 50, (ACC = *50), Hex = 32
- 1: 0100_110011 // ADD 51, ACC = ACC * (*51), Hex = 133
- 2: 0001_110100 // STO 52, (*52) = ACC, Hex = 74
- 3: 1001_000000 // Halt, Hex = 240
- 50: 0000000101 // Hex = 5
- 51: 0000001010 // Hex = A

FB-CPU

Örnek Yazılım 3

FB-CPU için bellekte 50 ve 51 adresteki iki sayının çarpımını 52 no'lu adrese kaydeden uygulamayı geliştiriniz. Ancak çarpma operasyonunu kullanmayınız. Çarpma işlemi için 50'deki sayıyı 51'deki sayı defa toplayıp 52 no'lu adrese yazınız. Gerekli değişkenler için istediğiniz adresleri kullanabilirsiniz.

- 0: 0000_110011 // LOD 51, ACC = *51, Hex = 33
- 1: 0011_110001 // SUB 49, ACC = ACC - *49, Hex = F1
- 2: 0111_001010 // JMZ 10, döngü bittiyse, döngüden çıkartacaktır (ACC-49 == 0), 10. Satır, Hex = 1CA
- 3: 0000_110000 // LOD 48, temp değerini yükle, başlangıçta 0, Hex = 30
- 4: 0010_110010 // ADD 50, ikinci sayıyı ACC'nin üstüne ekle, Hex = B2
- 5: 0001_110000 // STO 48, ACC'nin değerini temp'e ata, Hex = 70
- 6: 0000_110001 // LOD 49, ACC = i, Hex = 31

FB-CPU

Örnek Yazılım 3

- 7: 0010_101110 // ADD 46, ACC = i + 1, Hex = AE
- 8: 0001_110001 // STO 49, i = i + 1, Hex = 71
- 9: 0110_000000 // JMP 0, döngünün başına dön 0. satır, Hex = 180
- 10: 0000_110000 // LOD 48, ACC = temp, Hex = 30
- 11: 0001_110100 // STO 52, *52 = ACC, Hex = 74
- 10: 1001_000000// HLT, bitirme, Hex = 240
- 46: 1 // 1 sayısı
- 48: 0 // Hex = 0, temp
- 49: 0 // Hex = 0, i index'i için
- 50: 0000000101 // Hex = 5
- 51: 0000001010 // Hex = A

FB-CPU Teslimi

Proje Teslim Dokümanı ve Sunum

FB-CPU

FB-CPU Teslimi

- Proje Teslim Dokümanının, ders sınıfı ve saatinde, çıktılarının alınarak teslim edilmesi gerekmektedir.
- Ayrıca LMS'te açılmış olan "Proje Teslim" sayfasına aşağıdaki dosyaların yüklenmesi gerekmektedir.
- Logisim simulatoru üzerinde yapılan işlemci tasarımı (.circ uzantılı dosya)
- Makine dilindeki iki yazılım (.fbcpusoftware uzantılı dosyalar)
- Hazırlanan powerpoint sunum dosyası (.ppt uzantılı dosya)
- Proje Teslim Dokümanı (Word formatında yüklenmelidir) o Dokümanın alt başlıkları doldurulmalıdır
- Kaydedilen powerpoint sunum video'su youtube'a yüklenip, adresi, dokümanın sonuçlar bölümündeki açılmış yere link'i yazılmalıdır (Video'nun herkes'e görünür olmamasını istiyorsanız, youtube'a yükledikten sonra liste dışı seçeneğini seçerek, sadece link'e sahip olan kişilerin görmesini sağlayabilirsiniz).
- LMS'e yüklenen tüm dosyalar (Logisim işlemci tasarım dosyası, makine dilindeki iki yazılım, ppt uzantılı sunum dosyası ve Proje Teslim Dokümanını (PDF formatında)), github.com sitesine üye olup, yüklenip, dokümanın sonuçlar bölümündeki yere link'i yazılmalıdır.